VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi,Karnataka - 590 018.

A Project Work Report on

"DESIGN AND IMPLEMENTATION OF AMBA APB PROTOCOL USING 90nm TECHNOLOGY"

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

by

Pratheek T G	**USN:4JN21EC070**
Praveen V Makanur	**USN:4JN21EC073**
Raghu P R	**USN:4JN21EC075**
Adil Basha	**USN:4JN22EC401**

Under the Guidance of

Mrs.Shwetha B M.Tech.

Assistant Professor,

Dept. of ECE, JNNCE, Shimoga-577 204.

Department of Electronics and Communication Engineering

JNN College of Engineering, Shimoga - 577 204.

December 2024

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590 018.

JNN College of Engineering

Department of Electronics and Communication Engineering

Shimoga-577 204.

CERTIFICATE

This is to certify that the project work entitled **"DESIGN AND IMPLEMENTATION OF AMBA APB PROTOCOL USING 90nm TECHNOLOGY"** is carried out by **Pratheek T G (4JN21EC070), Praveen V Makanur(4JN21EC073), Raghu P R(4JN21EC075), Adil Basha(4JN22EC401)** , the bonafide students of JNN College of Engineering, Shimoga in partial fulfillment for the award of "Bachelor of Engineering" in department of "Electronics and Communication Engineering" of the Visvesvaraya Technological University, Belagavi, during the year 2024-25. It is certified that all the corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Signature of the Guide
Prof. Shwetha B
Assistant Professor,
Dept. of ECE,JNNCE, Shimoga.

Signature of the Coordinator
Prof. Ujwala B S
Assistant Professor
Dept. of ECE,JNNCE, Shimoga.

Signature of the HoD
Dr. S.V. Sathyanarayana
Professor & HoD
Dept. of ECE,JNNCE, Shimoga.

Signature of the Principal
Dr. Y. Vijaya Kumar
Principal
JNNCE, Shimoga.

External Viva

Name of the examiner — Signature with date

1.

2.

ABSTRACT

The increasing complexity of modern digital systems necessitates efficient bus protocols and optimized physical designs to meet performance and area constraints. This project presents a detailed study on the design and implementation of the AMBA Advanced Peripheral Bus (APB) protocol, integrating it with a 90nm technology node for physical design optimization. The objectives of this work include designing a reliable AMBA APB interface, verifying its functional correctness using Universal Verification Methodology (UVM), and executing a comprehensive physical design flow to achieve optimal performance, power, and area metrics.

A thorough literature survey on bus protocols and 90nm CMOS technology provides insights into minimizing latency and optimizing low-power designs for peripheral communication. The proposed methodology focuses on implementing the AMBA APB protocol, ensuring efficient communication between master and peripheral devices with minimal complexity. Verification is performed to validate timing, protocol compliance, and data integrity under various operational conditions.

The physical design flow incorporates critical steps such as floorplanning, power planning, layout optimization, and timing analysis, targeting reduced power consumption and enhanced area utilization. Results from post-physical design flow demonstrate significant improvements in timing closure, power efficiency, and design area, highlighting the efficacy of the 90nm technology node in implementing the AMBA APB protocol.

By combining theoretical insights from existing research with practical design strategies, this study contributes to the advancement of high-performance, low-power peripheral communication systems. The findings underscore the importance of thorough verification and physical design optimization to ensure the reliability and efficiency of bus-based digital systems.

Keywords: SoC, AMBA, APB, AXI, AHB, ASB, VLSI, design verification; Methodology, Physical Design Flow.;

ACKNOWLEDGEMENTS

We would like to acknowledge the help and encouragement given by various people during the course of this project. We would be thankful to our beloved principal for providing excellent academic climate. The support provided by the college and departmental library is greatly acknowledged. I thank to **Dr. Y. Vijaya Kumar**, Principal, JNNCE, Shivamogga for providing an excellent academic climate. We would like to express our sense of obligation to **Dr. P Manjunath** Professor and Dean academics, J. N. N. College of Engineering, Shivamogga, for his kind support, guidance and encouragement throughout the course of this work. I would like to express our sincere gratitude to **Dr. S.V. Sathyanarayana**, Dean RI&D, Head of Department, of Electronics and Communication Engineering, J. N. N. College of Engineering, Shivamogga, for his kind support, guidance and encouragement throughout the course of this work. I would like to express our grateful thanks to Major Project Coordinator **Mrs. Ujwala B S**, Assistant Professor, Department of Electronics and Communication Engineering, J. N. N. College of Engineering, Shivamogga, for his kind support, guidance and encouragement throughout the course of this work. I was deeply indebted to the invaluable guidance given by **Mrs. Shwetha B** , Assistant Professor Department of Electronics and Communication Engineering, J.N.N College of Engineering, Shivamogga, during this project work. I would like to thank all the teaching and non-teaching staff of Dept. Of ECE for their kind cooperation during the course of the work. Finally, I thankful to our parents and friends, who helped us in one way or the other throughout our project work.

Pratheek T G

Praveen V Makanur

Raghu P R

Adil Basha

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The AMBA APB (Advanced Peripheral Bus) protocol is a low-power, low-complexity interface for connecting peripherals in digital systems. As part of ARM's Advanced Microcontroller Bus Architecture (AMBA), it facilitates efficient data transfer between the CPU or other master modules and slower peripherals such as UART, I2C, and GPIO. The APB's simplicity and non-pipelined structure make it ideal for applications where high bandwidth is not required, ensuring robust performance with minimal design overhead.

In this project, the AMBA APB module is implemented at the RTL level using Verilog, targeting 90nm technology. The design features parameterized configurations for scalable address and data widths, ensuring compatibility across different systems. Key protocol signals such as PSEL, PENABLE, PWRITE, PADDR, PWDATA, and PRDATA are incorporated to enable seamless communication while maintaining compliance with the AMBA specification. Additionally, the design optimizes for low power consumption, adhering to timing and electrical constraints of 90nm technology, making it a reliable and efficient solution for modern embedded systems.

1.1 Objectives:

1. To design a Configurable AMBA APB Protocol
2. To stimulate the AMBA APB Protocol design using Xcelium.
3. To generate basic synthesis flow for the design using genus
4. To run the implementation flow including floor planning , placement, power planning and routing
5. To clean up DRC and LVS on the design with best results

1.2 Methodology:

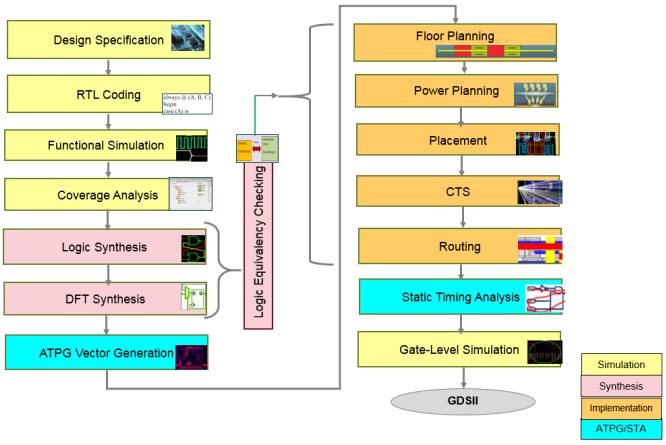

Figure 1.1: RTL to GDSII Design flow

- **Design Specification :**The design begins with defining core parameters, including data width (32-bit or 64-bit) and operational constraints within a single clock domain. The APB protocol features simplified control signals (PCLK, PRESETn, PSEL, PENABLE, PWRITE, PADDR, PWDATA, and PRDATA) and a streamlined handshake mechanism. It encompasses functional blocks such as the master and slave interfaces, which manage address decoding and data transactions. Verification involves a comprehensive testbench using UVM, and the design methodology leverages SystemVerilog with tools like QuestaSim for simulation. The implementation in 90nm CMOS technology aims to balance performance, power consumption, and area, incorporating power-saving modes to enhance efficiency.

- **RTL Design :**The RTL design of the AMBA APB protocol starts with defining specifications such as bus width, clock frequency, and single-clock domain operation. Simplified control signals (PCLK, PRESETn, PSEL, PENABLE, PWRITE, PADDR, PWDATA, and PRDATA) are used to enable efficient read and write transactions. The design includes modular master and slave interfaces for transaction initiation, address decoding, and data handling. Hierarchical design ensures scalability and ease of integration. Power optimization techniques are applied to meet the constraints of 90nm CMOS technology, balancing performance and efficiency. Verification is performed using Verilog and simulation tools like Cadence, ensuring compliance and robustness. The final design is synthesizable, reliable, and ready for implementation.

- **Functional Verification :** Functional verification ensures the correctness of the RTL design. A comprehensive testbench is developed to simulate scenarios like normal operation, boundary cases (full and empty conditions), and stress tests for handling clock domain crossings. Simulations are run using Cadence Xcelium,generating waveforms and reports for debugging. Assertions are used to validate critical conditions, and coverage metrics (code and functional) are measured to ensure thorough testing. Achieving high coverage (90) is crucial to identifying any potential design flaws before synthesis.

- **Logic Synthesis :**Logic synthesis, performed using Cadence Genus, converts the verified RTL into a gate-level netlist while adhering to timing, area, and power constraints. Constraints like clock definitions, multi-cycle paths, and false paths for asynchronous crossings are specified in the synthesis process. The tool optimizes the logic, removes redundancies, and maps the design to a standard cell library. The synthesis report provides insights into timing violations, area utilization, and power estimates, which are addressed before moving to the next stage.

- **Static Timing Analysis (STA) :**Static timing analysis ensures the design meets timing requirements across all paths, especially critical asynchronous crossings. Cadence Tempus is used to verify setup and hold times for both clock domains and check for violations. Timing paths like clock domain crossings are analyzed and constraints are refined to eliminate violations. This step is critical to ensuring that the design operates reliably under the specified clock frequencies.

- **Design for Testability (DFT) :** DFT techniques are incorporated to facilitate testing after fabrication. Scan chains are added for easier debugging, and proper reset mechanisms for both clock domains are implemented. Tools within the Ca dence suite are used to verify the design's testability and ensure that it meets DFT requirements without compromising functionality or timing.

- **Physical Design :**Physical design involves multiple steps, starting with floorplan ning and placement using Cadence Innovus. The memory macros and standard cells are placed efficiently to minimize area and routing congestion. Clock tree synthesis (CTS) ensures that clock skew is minimized within each domain while maintaining separation between write and read clocks. Routing is performed to interconnect all components, resolving congestion and fan-out issues while meet

ing timing constraints. Power optimization techniques, such as clock gating, are applied to reduce dynamic power consumption.

- **Post-Layout Verification :**Post-layout verification ensures that the physical de sign matches the original RTL intent. Static timing analysis is rerun using Tempus to verify timing closure after placement and routing. Physical verification, includ ing DRC (Design Rule Check) and LVS (Layout Versus Schematic), is performed using tools like Cadence Pegasus or Assura to ensure manufacturability and cor rectness. Power analysis is conducted to validate power dissipation under typical and worst-case scenarios.

- **Signoff and GDSII Generation:**The final step is signoff, where all timing, func tional, and physical checks are reviewed to ensure compliance with the design spec ifications. Any remaining violations are resolved, and the design is finalized. The GDSII file is then generated using Cadence Innovus for tape-out, representing the physical layout ready for fabrication. This step marks the completion of the asyn chronous FIFO design process from RTL to GDSII.

Chapter 2

Literature Survey

1. **Vaishnavi, R. K., S. Bindu, and Sheik Chandbasha. "Design and Verification of APB Protocol by using System Verilog and Universal Verification Methodology." International Research Journal of Engineering and Technology (IRJET) 6, no. 06 (2019): 23950072.**

   The paper "Design and Verification of APB Protocol using System Verilog and Universal Verification Methodology" by Vaishnavi R.K, Bindu S., and Sheik Chand Basha, published in June 2019 in IRJET, explores the design and verification of the Advanced Peripheral Bus (APB) protocol within the AMBA architecture. The study begins with a detailed analysis of the APB protocol, emphasizing its simplicity and efficiency for low-bandwidth peripheral communication. The APB design is implemented using Verilog HDL, ensuring compliance with standard protocol specifications. Verification is carried out using System Verilog and Universal Verification Methodology (UVM), which provides a structured, reusable testbench environment for thorough validation of the bus. The results demonstrate error-free data transactions, ensuring data integrity during read and write operations. The paper underscores the benefits of leveraging Verilog for design and UVM for verification, achieving a robust and reliable APB protocol implementation suitable for modern low-bandwidth applications.

   Key findings- Includes the simplicity of the APB protocol, successful implementation using Verilog HDL, a structured verification environment with System Verilog and UVM, error-free data transactions, and the protocol's suitability for modern low-bandwidth applications.

2. **Bolla, Dileep Reddy, and Satya Srikanth Palle. "Design and Implementation of Advanced Peripheral Bus (APB) Protocol in Systemverilog." (2022).**

   The paper "Design and Implementation of Advanced Peripheral Bus (APB) Protocol in System Verilog"* by Dr. Dileep Reddy and Dr. Sathya Srikanth Palle, published in 2020, explores the APB protocol's crucial role in ARM-based embedded systems for efficient peripheral communication. The authors focus on implementing the protocol using System Verilog, taking advantage of its advanced features for both design and verification. The study provides a detailed breakdown of APB protocol components, including address phases, data phases, control signals, and transaction types, to ensure a comprehensive understanding of its operation. This paper uniquely emphasizes protocol optimization by refining design specifications to improve efficiency and reduce latency. System Verilog's strengths are highlighted in enhancing the design process, enabling robust and scalable hardware implementations. The research presents an accurate and functional APB protocol design with improved reliability, ensuring streamlined peripheral-to-CPU communication. **Key findings-** Includes the critical role of the APB protocol in ARM-based embedded systems, effective implementation using System Verilog with an emphasis on protocol optimization for reduced latency, and enhanced reliability in peripheral-to-CPU communication.

3. **Paunikar, Abhijeet, Rohan Gavankar, Nachiket Umarikar, and K. Sivasankaran. "Design and implementation of area efficient, low power AMBA-APB Bridge for SoC." In 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), pp. 1-6. IEEE, 2014.**

   The paper *"Design and Implementation of an Area-Efficient, Low-Power AMBA-APB Bridge for SoC"* by Panikkar A, Gavankar R, and Umarikar N, published in 2022, focuses on optimizing communication bridges within System-on-Chip (SoC) architectures. It addresses the challenge of integrating peripherals that use different bus protocols, such as AMBA and APB, while achieving area efficiency and low power consumption. Unlike previous works that concentrate solely on protocol implementation, this research emphasizes optimizing the bridge design to reduce power overheads and silicon area, which are critical for modern SoC designs. The

authors provide a thorough analysis of AMBA and APB architectures, emphasizing their role in efficient peripheral communication. Through innovative design strategies, the paper demonstrates substantial improvements in the bridge's performance, power usage, and area utilization, targeting hardware resource constraints and delivering an optimized solution for bus integration in complex SoC environments.

Key findings- Includes optimizing the AMBA-APB bridge design for reduced power consumption and silicon area, while ensuring efficient peripheral communication in System-on-Chip (SoC) architectures.

4. **Dwivedi, Anushka, and Anand Jatti. "DESIGN AND IMPLEMENTATION OF AMBA APB PROTOCOL USING SYSTEM VERILOG." International Research Journal of Modernization in Engineering Technology and Science 4, no. 07 (2022).**

   The paper "Design and Implementation of AMBA APB Protocol Using System Verilog" by Anushka Dwivedi and Dr. Anand Jatti (July 2022) focuses on the significance of the AMBA APB protocol in ARM-based systems, emphasizing its role in efficient peripheral-to-CPU communication. It highlights the simplicity and low-power nature of the APB protocol, making it suitable for connecting low-bandwidth peripherals. The authors introduce System Verilog as a powerful language for hardware description and verification, demonstrating its advantages in both design and verification. The implementation uses System Verilog's features to accurately model the APB protocol, ensuring compliance and performance efficiency. Verification methods are thoroughly discussed to validate the design across various scenarios, ensuring reliability and correctness. The study presents a robust framework for designing communication protocols using System Verilog, aimed at enhancing the performance and efficiency of advanced hardware systems.

   Key findings- The study found that System Verilog is an effective tool for designing and verifying AMBA APB protocols, enabling efficient and reliable peripheral-to-CPU communication in ARM-based systems. The approach effectively balances simplicity, power efficiency, and compliance with standard protocol requirements.

5. **Mukunthan, J., A. Daniel Raj, S. Keerthivadana, R. Kiruthika, T. Shruthi, and S. Snegasri. "Design And Implementation Of Amba Apb Protocol." In IOP Conference Series: Materials Science and Engineering, vol. 1084, no. 1, p. 012050. IOP Publishing, 2021.**

The paper "Design and Implementation of AMBA APB Protocol" by Mukunthan J, A Daniel Raj, and Shruthi T, published in 2021, focuses on the APB protocol's role in facilitating seamless communication between the CPU and peripherals in ARM-based embedded systems. The authors provide detailed specifications of the protocol, including transaction structure, addressing schemes, and control signals, to clearly explain APB operations. Unlike previous studies, this work emphasizes architectural design for implementing the APB protocol, presenting a modular and efficient approach that enhances scalability and integration ease. The proposed architecture optimizes transaction flow between master and slave, reducing communication latency. The paper highlights practical design considerations for reliable APB protocol implementation in resource-constrained systems. Overall, the research contributes to a robust and efficient design framework tailored specifically for embedded systems.

Key findings- This study presents an architectural approach that optimizes transaction flow and reduces latency in the APB protocol, making it particularly suitable for resource-constrained ARM-based embedded systems. The modular design enhances scalability and integration ease, providing a solid foundation for reliable APB protocol implementation.

Chapter 3

Theoretical Background

3.1 AMBA's bus architecture

The Advanced Microcontroller Bus Architecture (AMBA) comprises a suite of interconnect specifications developed by ARM (Advanced RISC Machines) to facilitate the creation of high performance, low-power systems-on-chip (SoC). Among its essential elements, the Advanced Peripheral Bus (APB) stands out, designed to furnish a cost-effective, energy-efficient interface for linking peripherals to the system. AMBA APB operates on a straightforward, low-frequency bus protocol, commonly utilized for interfacing various low-bandwidth peripherals like timers, interrupt controllers, and supplementary components to the primary system bus. Specifically crafted for applications where top-tier performance isn't imperative, APB proves ideal for connecting slower peripherals.

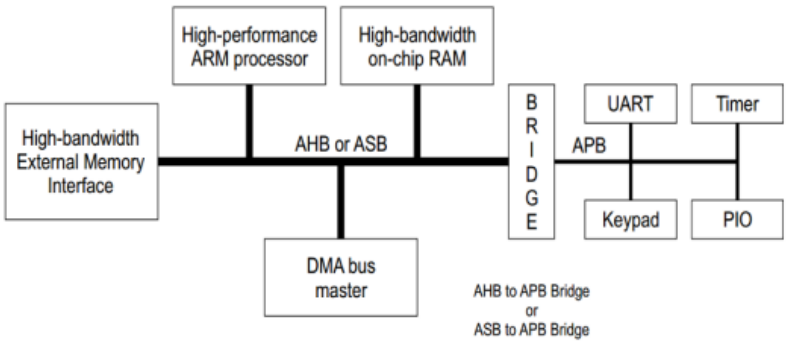

Figure 3.1: AMBA's bus architecture

AMBA encompasses a collection of interconnect protocols facilitating communication among diverse components within a computer system, including processors, peripherals, and memory. Illustrated in the diagram are an advanced ARM processor and high-bandwidth on-chip RAM, with a central bridge connecting the processor to the external memory interface. Various bridges exist, tailored to different bus types, such as the AHB to APB Bridge or ASB to APB Bridge. The Advanced High-performance Bus is engineered for swift, high-bandwidth transfers between masters and slaves, while the

Advanced System Bus serves as a high-performance bus primarily utilized for memory system interfaces. Contrarily, the Advanced Peripheral Bus caters to simpler peripherals not necessitating high-performance transfers and emphasizes low-power consumption. Notable peripherals include the UART for serial communication, Timer for generating regular electrical pulses, Keypad for data entry, PIO for byte-by-byte data transfer through a parallel port, and DMA, a technique enabling peripherals to transfer data directly to memory without CPU involvement.

3.1.1 APB protocol

The APB protocol stands out as a cost-effective interface, prioritizing minimal power usage and simplified interface structures. It operates on a non-pipelined, straightforward synchronous protocol, ensuring each transfer completes in at least two cycles. Widely adopted, the AMBA APB protocol serves as a prevalent standard for on-chip communication in system-on-chip (SoC) designs, especially prevalent in ARM-based architectures. Originating from ARM Holdings, the APB protocol streamlines communication among diverse components within an SoC, including CPUs, memory controllers, and peripheral devices.

The APB interface is tailored for accessing the programmable control registers found in peripheral devices. Usually, APB peripherals link to the primary memory system via an APB bridge. For instance, an AXI to APB bridge facilitates the connection of multiple APB peripherals to an AXI memory system. Initiating APB transfers falls under the responsibility of an APB bridge, also known as a Requester. Meanwhile, a peripheral interface acknowledges requests, alternatively known as a Completer.

This specification will employ the terms Requester and Completer. Transactions in the APB protocol consist of address and data phases, where the master device asserts control signals to indicate the start and end of each transaction. The protocol supports single cycle transfers, allowing for relatively fast communication between components. One notable aspect of the APB protocol is its ease of integration into SoC designs, enabling efficient utilization of resources and minimizing design complexity. Additionally, its widespread adoption within ARM based systems ensures compatibility and interoperability among various hardware components and peripherals the APB protocol include simplicity, low complexity, and efficiency, making it suitable for connecting peripheral devices with moderate bandwidth requirements. It operates as a simple master/slave

protocol, where a master device initiates transactions to read from or write to slave devices.

3.1.2 APB Master and APB Slave

- With only one bus master on the APB, an arbiter becomes unnecessary.
- The master oversees both the address and write buses, performing a combinatorial decode of the address to decide which PSELx signal to activate.
- Additionally, it manages the timing of the transfer by controlling the PENABLE signal.
- In a read transfer, it transmits APB data onto the system bus.

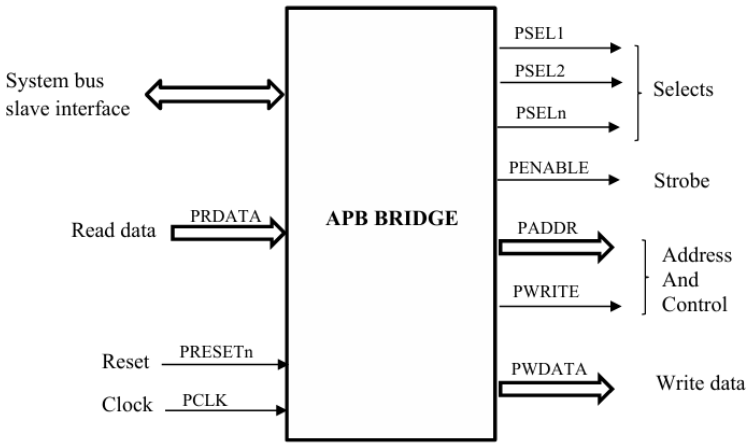

Figure 3.2: APB Master

- The interface of APB slaves is characterized by its simplicity and flexibility.
- The specific configuration of this interface will vary based on the chosen design approach, allowing for numerous potential options.
- Two crucial signals in this interface, PSLVERR and PREADY, primarily ensure data integrity during data transfer processes.

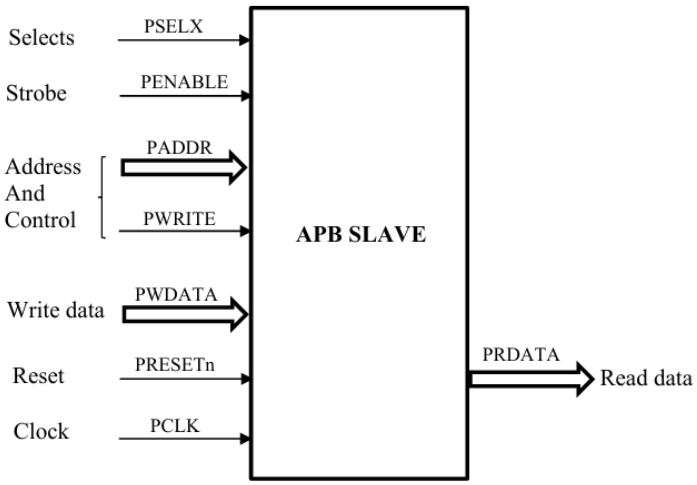

Figure 3.3: APB Slave

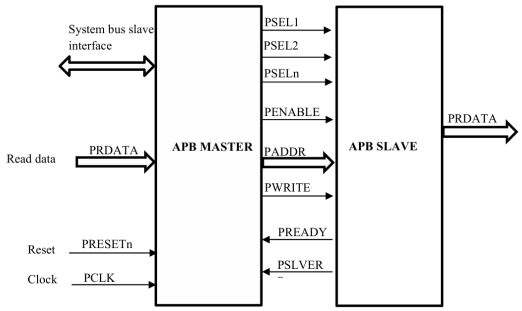

Figure 3.4: Communication of Master to Slave

3.1.3 Communication of Master to Slave

- **System bus slave interface:** The interface connecting the peripheral device to the system bus.
- **PCLK:** Symbolizing the system clock.
- **PRESETn:** An active-low asynchronous reset signal.
- **PADDR:** Signifying the address bus from the master to the slave, width of up to 32 bits.
- **PWDATA:** The write data bus from the master to the slave, potentially up to 32 bits wide.
- **PRDATA:** The read data bus from the slave to the master, with a potential width of up to 32 bits.
- **PSELx:** Slave select signals, with one PSEL signal designated for each slave linked to the master.

- **PENABLE:** The enable signal for the data bus.
- **PWRITE:** Indicating a write transfer.
- **PRREADY:** Utilized by the slave to signify readiness to receive data during a write transfer.
- **PSLVER:** Employed by the slave to confirm the validity of data during a read transfer. These signals facilitate communication between master and slave devices. To initiate a data transfer, the master places the address and data (for a write transfer) on the bus while asserting the necessary control signals. Upon recognizing its address on the PADDR bus, the slave device responds by asserting the PREADY signal, indicating readiness to receive data. Once the data is received, the slave asserts the PSLVER signal to confirm the validity of data on the PRDATA bus. Subsequently, the master retrieves the data from the PRDATA bus.

3.2 Operational States

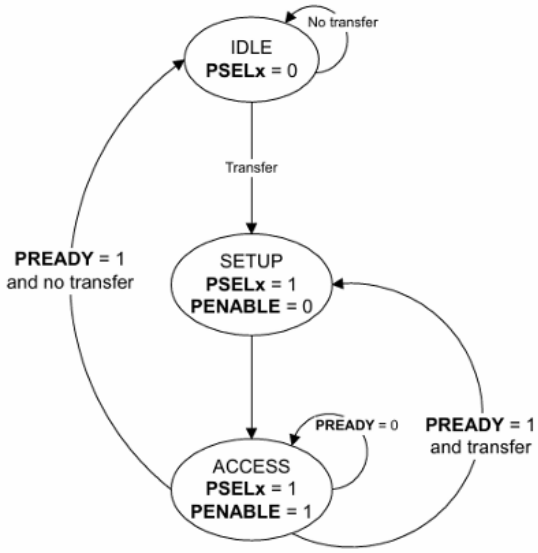

Figure 3.5: Diagram depicting States

The state machine progresses through the following states:

- **IDLE:** This represents the default state of the APB interface.
- **SETUP:** When a transfer is initiated, the interface transitions into the SETUP state. Here, the relevant select signal, PSELx, is activated. The interface stays in

the SETUP state for only one clock cycle and invariably transitions to the ACCESS state on the subsequent rising clock edge.

- **ACCESS:** In the ACCESS state, the enable signal, PENABLE, is activated. It's crucial that the following signals remain unchanged during the transition from SETUP to ACCESS and between cycles within the ACCESS state:
  - **PADDR**
  - **PPROT**
  - **PWRITE**
  - **PWDATA**, only for write transactions
  - **PSTRB**
  - **PAUSER**
  - **PWUSER**

  The ACCESS state termination is regulated by the PREADY signal originating from the Completer

- Should the Completer maintain PREADY at a LOW level, the interface persists in the ACCESS state.

- Conversely, if the Completer sets PREADY to a HIGH state, the ACCESS state concludes. If further transfers are unnecessary, the bus reverts to the IDLE state. Alternatively, if another transfer is imminent, the bus transitions directly to the SETUP state.

3.3 Transfers

Write transfers

1. **Transfer without any wait states:** The initiation of the write transfer occurs

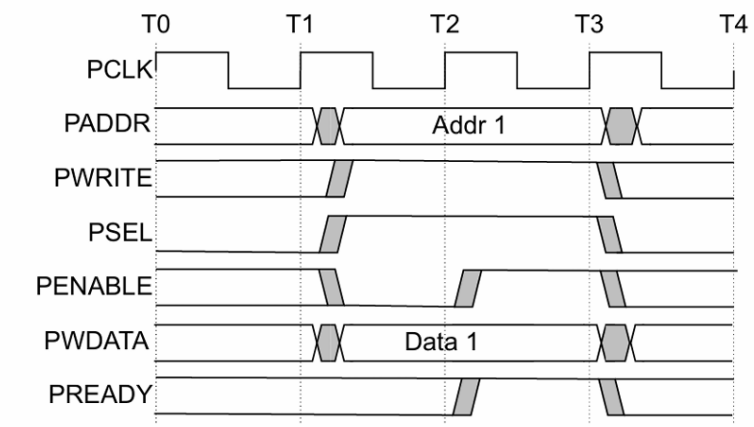

Figure 3.6: Transfer without wait states

when the address, write data, write signal, and select signal all undergo changes subsequent to the rising edge of the clock.

• The initial clock cycle of the transfer, termed the setup phase, ensues.

• Subsequent to the ensuing clock edge, the enable signal PENABLE is asserted, signaling the commencement of the Access phase.

• Throughout the Access phase, the address, data, and control signals remain valid, culminating in the completion of the transfer.

• At the termination of the transfer, PENABLE is de-asserted.

• PSELx transitions to a LOW state unless the transfer is promptly succeeded by another transfer to the same peripheral.

• **T1:** The master asserts all the necessary signals to initiate the write transfer. These signals include:

- PCLK: System clock signal.
- PADDR: Address bus that specifies the memory location where the data needs to be written.
- PWRITE: Write control signal (active high).
- PSEL: Slave select signal to identify the specific slave device involved in the transfer.

- PENABLE: Enable signal for the data bus.
- PWDATA: Write data bus that carries the data to be written (Data 1).

• T2: The slave asserts the PREADY signal. This indicates to the master that the slave is ready to accept the data on the PWDATA bus. Since PREADY is asserted in the same clock cycle (T2) as the other control signals by the master, there are no wait states inserted by the slave device in this transfer.
• T3: The write transfer is complete. The master deasserts PENABLE and PSEL signals.

Read transfers

2. Transfer without any wait states

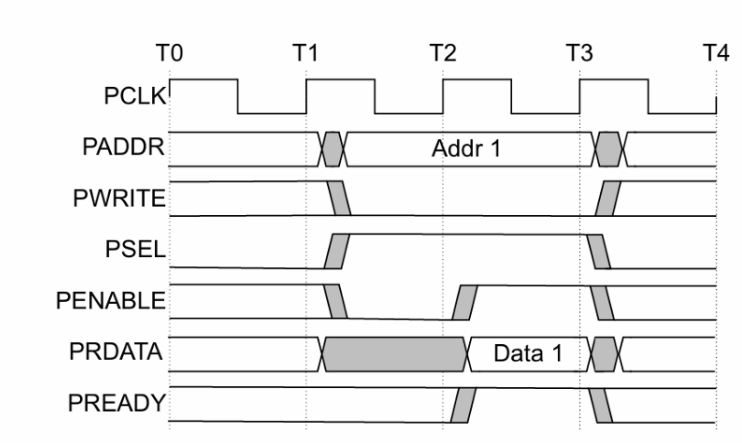

Figure 3.7: Transfer without wait states

The initiation of the read transfer occurs when the address, write signal, and select signal undergo changes following the rising edge of the clock.
• The initial clock cycle of the transfer, known as the Setup phase, follows.
• Subsequent to the ensuing clock edge, the assertion of PENABLE indicates the commencement of the Access phase.
• Throughout the Access Phase, the address and control signals retain their validity.
• It is imperative for the slave to furnish the data before the conclusion of the read transfer.
• The transfer reaches completion at the end of this cycle.
• PENABLE is de-asserted upon the conclusion of the transfer.

• **T0:** The master asserts all the necessary signals to initiate the read transfer. These signals include:

•PCLK: System clock signal

• PADDR: Address bus that specifies the memory location from where the data needs to be read (Addr 1).

• PWRITE: Write control signal (active LOW for a read transfer).

• PSEL: Slave select signal to identify the specific slave device involved in the transfer.

• PENABLE: Enable signal for the data bus.

• **T1:** The slave activates the PREADY signal simultaneously with transmitting Data 1 on the PRDATA bus. This signals to the master that the slave is prepared to deliver the data. As PREADY is asserted concurrently with the other control signals by the master in the same clock cycle (T1), there are no wait states introduced by the slave device in this transfer.

• **T2:** Upon completion of the read transfer, the master retrieves Data 1 from the PRDATA bus and removes the assertions of the PENABLE and PSEL signals.

This transfer method is optimal for high-performance systems where reducing transfer times is crucial. Ensuring that the slave device has the data readily available is essential to prevent reading outdated or inaccurate data. The capability to execute read transfers without introducing wait states relies on the functionalities of the individual slave device.

Chapter 4

Design and Implemetation

The design and implementation of the AMBA APB protocol involve creating a simple, low-power bus for peripheral communication. The design includes an APB Master (generating address, control, and data signals) and Slave (handling responses). Using Verilog, the RTL design is synthesized targeting 90nm CMOS technology. Innovus is employed for physical design, covering floorplanning, placement, CTS, routing, and optimization. Timing closure and power analysis are performed to meet 90nm constraints, followed by post-layout verification to ensure functionality and performance.

4.1 Design And Implemetation

The AMBA (Advanced Microcontroller Bus Architecture) APB (Advanced Peripheral Bus) protocol facilitates low-power, low-complexity communication for peripheral devices. To design and implement the APB protocol using 90nm technology, follow these steps:

4.1.1 Design And Implemetation Flow:

1. **Design Specification**

   The AMBA APB is used for low-power peripherals in an SoC. It operates with:

   - A single-clock domain.
   - Simple control signals (Address, Write/Read, Enable).
   - Handshaking between master and slave.
   - Sequential transfer behavior.

   The APB consists of two main components:

   - APB Master: Controls bus transactions (addressing and data transfer).
   - APB Slave: Responds to read/write requests from the master.

2. **RTL Design:** The design is implemented in Verilog. Key modules include:

   (a) Master Logic: Generates PADDR (address), PWRITE (write enable), and PWDATA (write data).

   (b) Slave Logic: Responds with PRDATA (read data) and PREADY (ready signal).

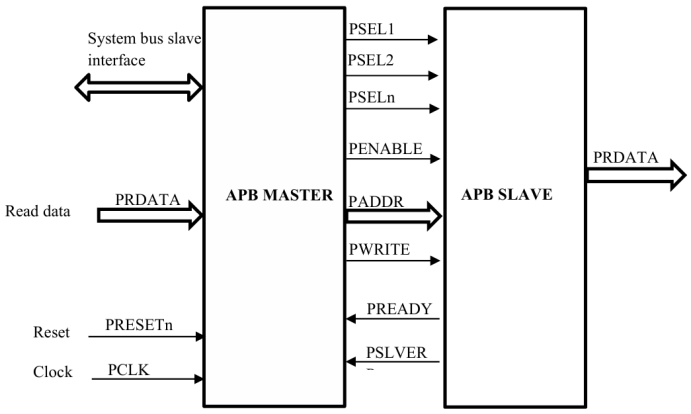

Figure 4.1: Communication of Master to Slave

3. **RTL Verification:**

   Simulate the design using a testbench in tools like ModelSim or Xilinx Vivado to ensure functionality. Use test cases for:

   - Single read/write transactions.
   - Burst transactions.
   - Response timing.

4. **Synthesis:**

   (a) Import the Verilog code into Synopsys Design Compiler or a similar tool.

   (b) Target the 90nm standard cell library.

   (c) Generate a gate-level netlist.

   (d) Perform timing optimization and verify constraints using STA (Static Timing Analysis).

5. **Physical Design (Using Innovus)**

(a) **Floorplanning:**

- Define core area and IO ring.
- Place macros for APB master and slave modules.

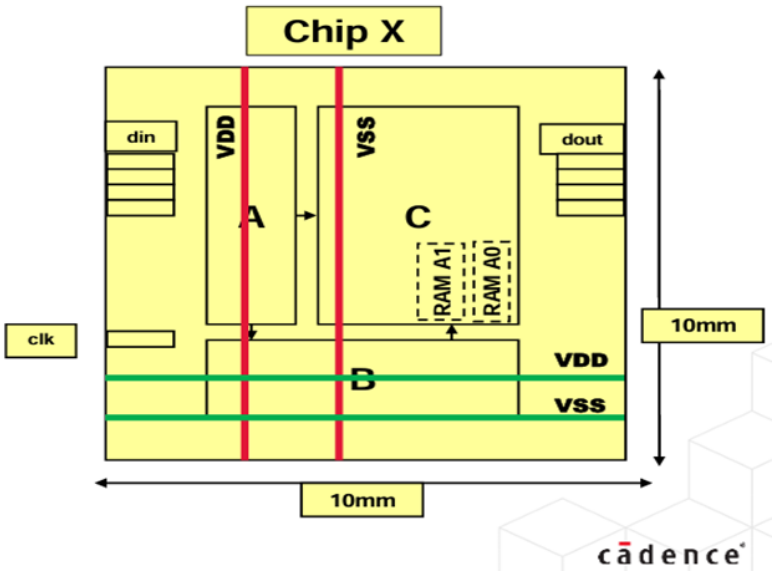

Figure 4.2: Floorplanning

(b) **Placement:**

- Place standard cells for logic gates generated during synthesis.
- Use Innovus Placement Engine for congestion-free placement

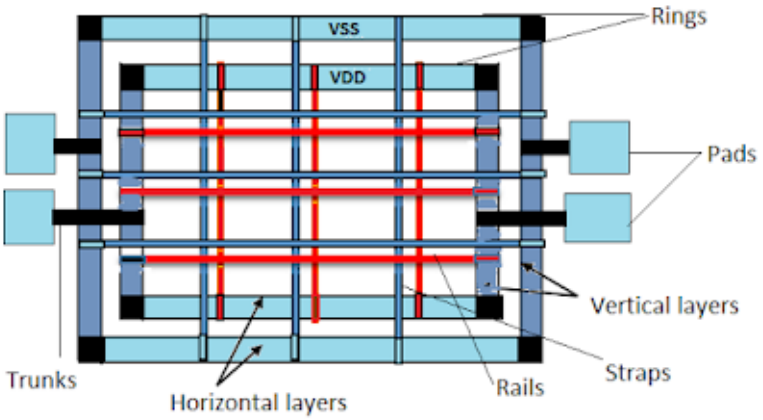

Figure 4.3: Placement:

(c) **Clock Tree Synthesis (CTS):**

- Design the clock tree to minimize skew and meet timing constraints.

- APB uses a single clock signal, simplifying CTS.

(d) **Routing:**

- Perform global and detailed routing.
- Ensure signal integrity by minimizing cross-talk and RC delays

(e) **Power Analysis:**

- Analyze power using Innovus' tools to ensure the design meets low-power requirements.

(f) **Sign-off:**

- Verify post-layout STA, DRC (Design Rule Check), and LVS (Layout vs. Schematic).

Chapter 5

Results and Analysis

5.1 Functional Simulation

After the RTL is coded, we will simulate the design using Cadence simulation tool called Xcelium. The functional simulation result includes

- Verified the correct operation of AMBA APB protocol signals, including proper handshaking between PSEL, PENABLE, and PREADY, and accurate transaction sequencing.
- Ensured accurate data transfer and integrity during read and write operations, with no data corruption and correct address decoding.
- Post-Layout Simulation Used Innovus with a 90nm process library to verify timing, power, and layout-level optimizations for performance and efficiency

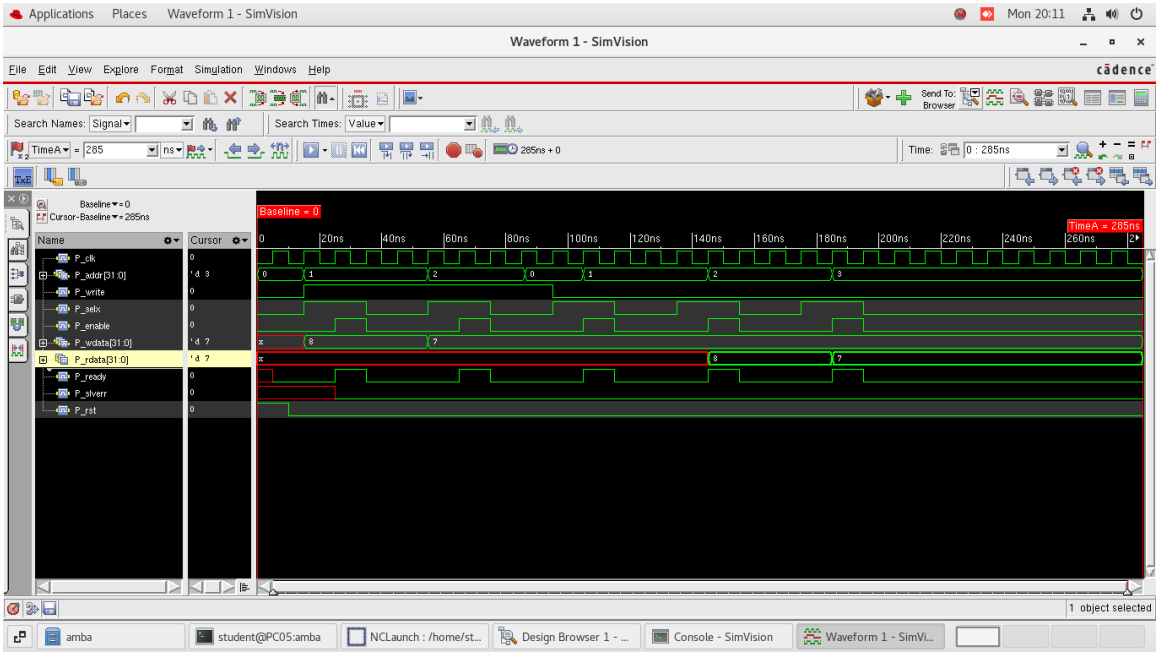

Figure 5.1: Simulated waveform

5.2 Synthesis (RTL to Gate-Level)

The RTL code is synthesized into a gate-level netlist using Cadence Genus. This process will:

- Map the RTL to standard cells in the target technology (cell libraries).
- Optimize for area, power, and timing.
- Perform static timing analysis (STA) to ensure the design meets timing constraints.
- Gate-level netlist ready for further stages.
- No timing violations after optimization (critical path must meet the clock speed).
- Low power and area-efficient design if constraints are met.

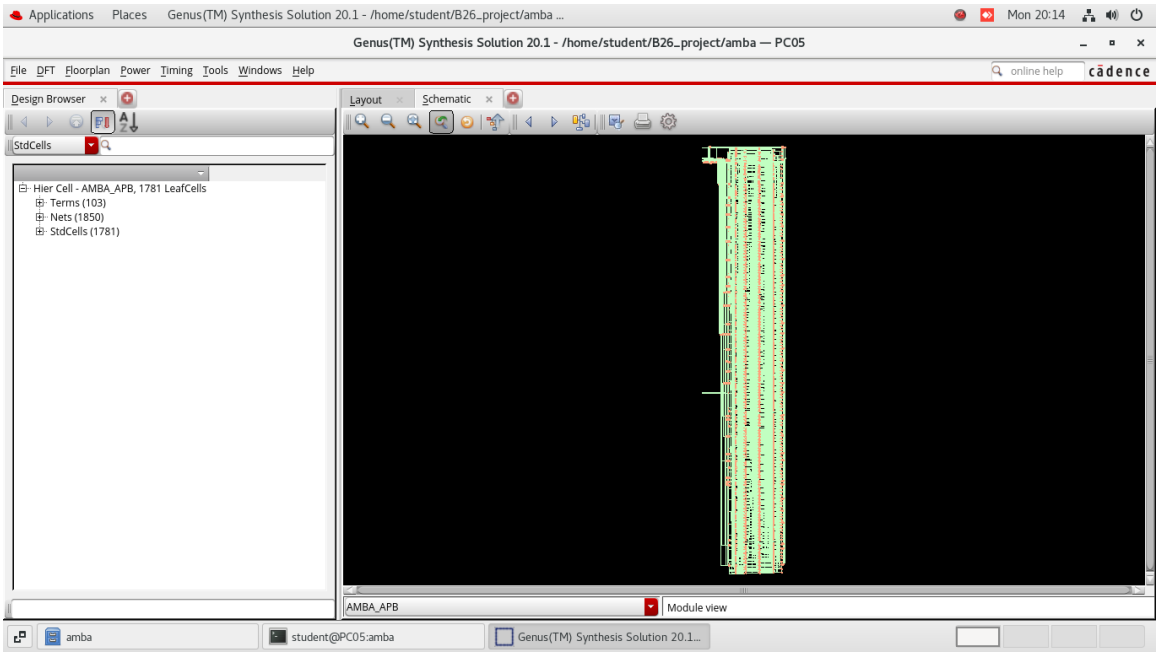

Figure 5.2: Synthesis

5.3 Place and Route

The gate-level netlist is used for placement and routing, with Cadence Innovus:

- **Placement:** The cells are placed within the chip area while respecting area constraints.
- **Routing:** The signal paths are routed between cells, ensuring minimal congestion and proper signal integrity.
- **Power Analysis:** Verifying that power consumption is within the acceptable limits.
- **DRC (Design Rule Check):** No violations related to the foundry's design rules.

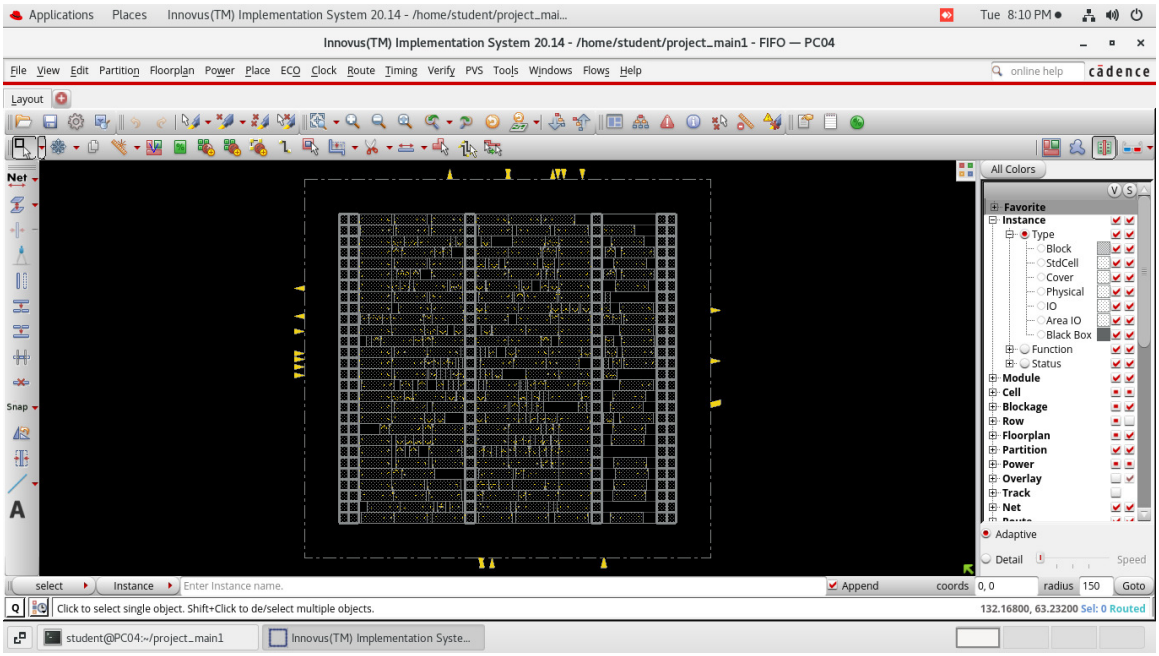

Figure 5.3: standard cell layout

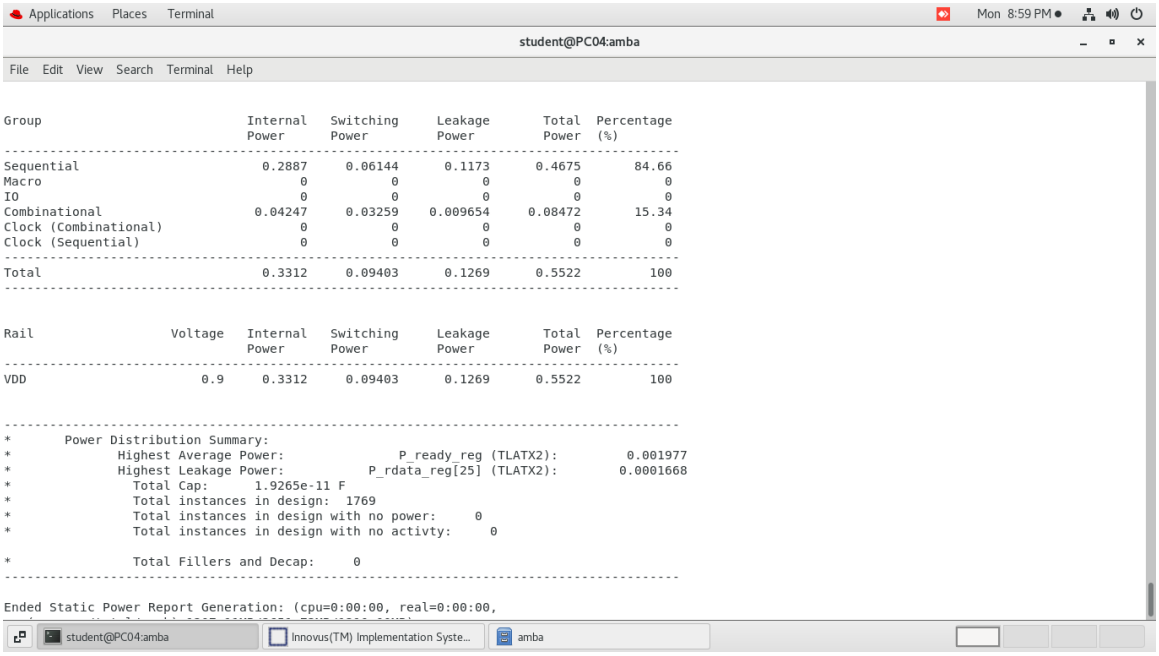

Figure 5.4: standard cell layout

5.4 Final Layout (GDSII Export):

Once all checks pass, the final layout is exported to GDSII format. This file contains all the physical layer data and is used for tape-out (sending the design to fabrication).

- The GDSII file includes all the geometric shapes representing transistors, wires, and other components on the chip.
- No errors in physical verification (DRC/LVS), and the design meets power, area, and timing requirements.

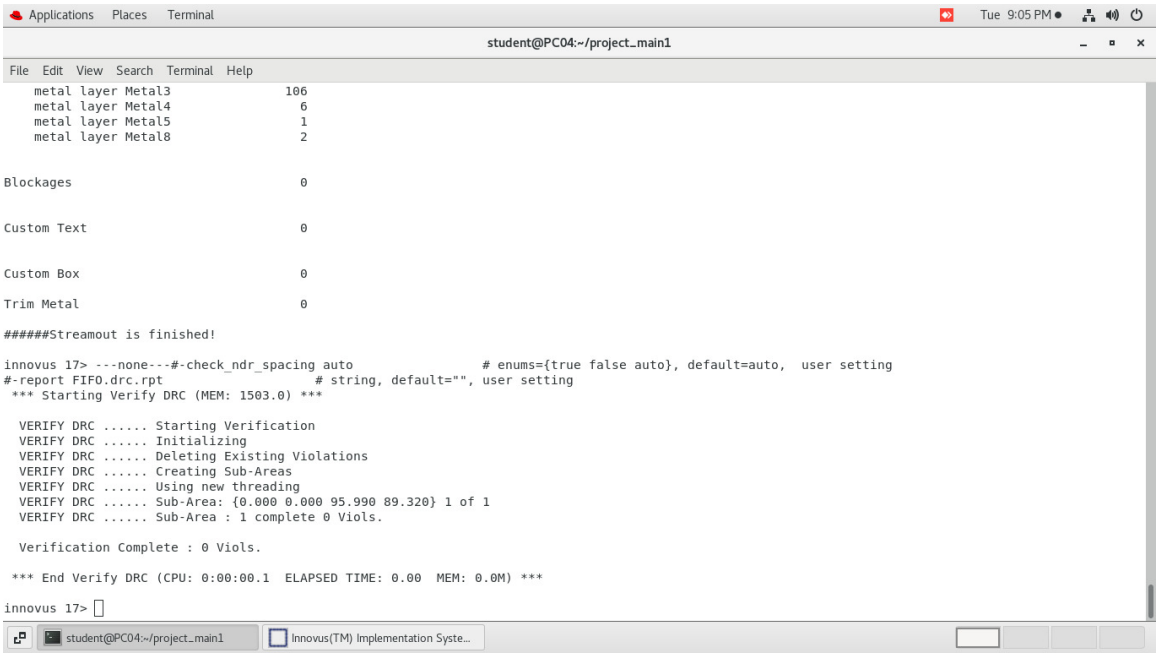

Figure 5.5: DRC verification

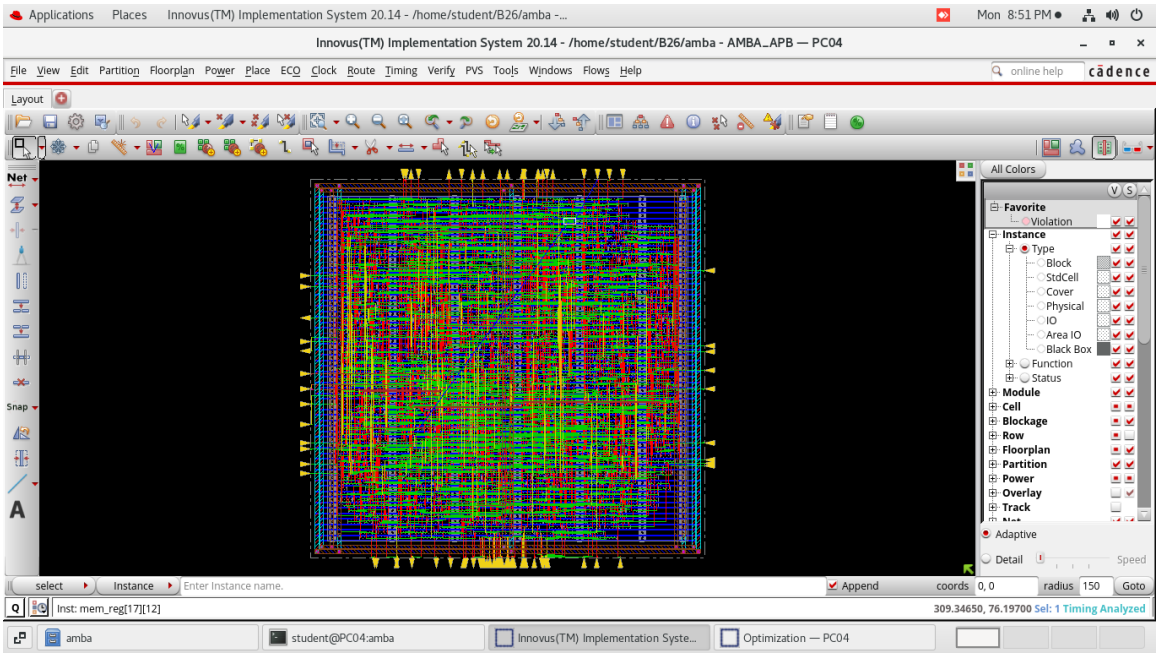

Figure 5.6: Final Layout

References

1. Vaishnavi R.K, Bindu S., and Sheik Chand Basha (2019): "Design and Verification of APB Protocol using System Verilog and Universal Verification Methodology," IRJET. The paper highlights APB protocol implementation using Verilog HDL and UVM, showcasing a reusable testbench for verifying error-free data transactions.

2. Dr. Dileep Reddy and Dr. Sathya Srikanth Palle (2020): "Design and Implementation of Advanced Peripheral Bus (APB) Protocol in System Verilog." The research focuses on System Verilog-based implementation, emphasizing protocol optimization for efficiency and reduced latency in ARM-based systems.

3. Panikkar A, Gavankar R, and Umarikar N (2022): "Design and Implementation of an Area Efficient, Low-Power AMBA-APB Bridge for SoC." This paper optimizes the APB bridge design, targeting low power and area efficiency for modern System-on-Chip (SoC) architectures.

4. Anushka Dwivedi and Dr. Anand Jatti (2022): "Design and Implementation of AMBA APB Protocol Using System Verilog." The study demonstrates the advantages of System Verilog in accurately modeling and verifying the APB protocol for reliable peripheral-to-CPU communication.

5. Mukunthan J, A Daniel Raj, and Shruthi T (2021): "Design and Implementation of AMBA APB Protocol." The work emphasizes modular architectural design to optimize transaction flow, scalability, and communication latency for embedded systems.